Estructuras de Datos 1 - ST0245
Segundo Parcial Grupo 032 (Martes)

Nombre .....
Departamento de Informática y Sistemas
Universidad EAFIT

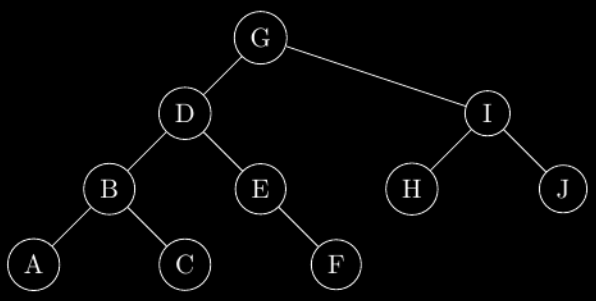
Mayo 07 de 2019

En las preguntas de selección múltiple, una respuesta incorrecta tendrá una deducción de 0.2 puntos en la nota final. Si dejas la pregunta sin responder, la nota será de 0.0. Si no conoces la respuesta, no adivines.

Para propósitos de este parcial, se considerará esta implementación de un árbol binario y un recorrido.

```
1  class BNode{ //Arbol binario
2    BNode izq;
3    BNode der;
4    int val;
5  }
6  void preorden(BNode nodo){ //Recorrido
7    if(nodo != null){
8      System.out.println(nodo.val);
9      preorden(nodo.izq);
10     preorden(nodo.der);
11   }
12 }
```

1 Árboles binarios de búsq. 30%



Dada la siguiente lista de enteros, ubica los números en el árbol anterior, de tal manera que el árbol final sea un árbol binario de búsqueda: $7, 9, 10, 8, 4, 5, 6, 2, 1, 3$

a) (10%) ¿Cuáles elementos van en cada letra si se insertan en el orden dado anteriormente?

i) $A = 1, B = 2, C = 3, D = 4, E = 5, F = 6, G = 7, H = 8, I = 9, J = 10$.

ii) $A = 4, B = 5, C = 6, D = 7, E = 8, F = 9, G = 1, H = 10, I = 2, J = 3$.

iii) $A = 7, B = 8, C = 9, D = 10, E = 1, F = 2, G = 3, H = 4, I = 5, J = 6$.

iv) $A = 7, B = 8, C = 9, D = 10, E = 1, F = 2, G = 3, H = 4, I = 5, J = 6$.

b) (10%) ¿Cuál sería el recorrido pre-orden del árbol anterior, asumiendo que siempre se visita primero la hoja de la izquierda?

i) $G, D, B, A, C, E, F, I, H, J$

ii) $A, B, C, D, E, F, G, H, I, J$

iii) $B, C, D, E, F, G, H, I, J, A$

iv) $A, C, B, D, I, J, H, E, F, G$

c) (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de encontrar una clave (número) en un árbol binario de búsqueda de n nodos?

i) $O(n)$

ii) $O(\log(n))$

iii) $O(n^2)$

iv) $O(1)$

2 Pilas 20%

El método `push(i)` ingresa el elemento i al tope de la pila. El método `pop()` retira el elemento en el tope de la pila y retorna su valor. Considere el siguiente método:

```
1  void metodo(Stack<Integer> s){
2    for(int i = 10; i >= 0; i--){
3      if(i % 2 == 0){
4        s.push(i);
5      }
6    }
7    System.out.print(s.pop());
8    while(s.size() > 0){
9      System.out.print(" " + s.pop());
10   }
11 }
```

a (10%) ¿Cuál es la salida del método anterior?

(i) 10, 8, 6, 5, 4, 2, 0

(ii) 2, 4, 6, 8, 10

(iii) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

(iv) 0, 2, 4, 6, 8, 10

b (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de añadir un nuevo elemento al tope de una pila que contiene n elementos?

(i) $O(1)$

(ii) $O(n)$

(iii) $O(n \log n)$

(iv) $O(n^2)$

3 Colas 10%

El método `add(i)` agrega el elemento i al inicio de la cola. El método `poll()` retira el elemento al final de la cola y retorna su valor. Considere el siguiente método:

```
1 void metodo(Queue<Integer> q){
2   for(int i = 2; i * i <= 25; i++){
3     q.add(i);
4   }
5   System.out.print(q.poll());
6   while(q.size() > 0){
7     System.out.print(" " + q.poll());
8   }
9 }
```

a (10%) ¿Cuál es la salida del algoritmo anterior?

(i) Los enteros positivos menores que 26 en orden ascendente.

(ii) Los enteros positivos menores que 26 en orden descendente.

(iii) 2, 3, 4, 5

(iv) 5, 4, 3, 2

4 Arboles N-arios 20%

La siguiente es una implementación de árbol n-ario.

```
1  class Nodo {
2    public int id;
3    public int valor;
4    public LinkedList<Nodo> hijos;
5  }
```

Hoy, Liko y Kefa, en clase de Estructura de Datos 1, aprendieron que con los árboles n-arios se pueden hacer muchísimas operaciones, tales como encontrar el tamaño de un sub-árbol, calcular la suma de los elementos en un sub-árbol, etc. Muchas de estas operaciones son realmente fáciles para Liko; sin embargo, hay una –en especial– difícil para Kefa. Para Kefa es difícil escribir un programa que calcule la suma de los elementos de los sub-árboles. Liko ha escrito un programa para calcular la suma de los elementos en los sub-árboles para enseñarle a Kefa e intencionalmente ha omitido una parte para asegurarse de que, si Kefa entiende el código, él sea capaz de encontrar el error. ¿Podrías ayudarles a Kefa a encontrar el error, por favor? Para propósitos de este ejercicio, Liko se ha asegurado que el árbol dado tiene exactamente n nodos, y que al método `suma` se le pasa la raíz del árbol y un arreglo con n ceros.

```
1  void suma(Nodo raiz, int[] suma){
2    if(raiz.hijos == null || raiz.hijos
         .size() == 0){
3      suma[raiz.id] = raiz.valor;
4      return;
5    }
6    suma[raiz.id] = raiz.valor;
7
8    for(Nodo e: raiz.hijos){
9      suma(e, suma);
10     suma[.........] = suma[e.id] +
           suma[raiz.id];
11   }
12 }
```

a) (10%) Completa, por favor, la línea 10 .....

b) (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, del código anterior?

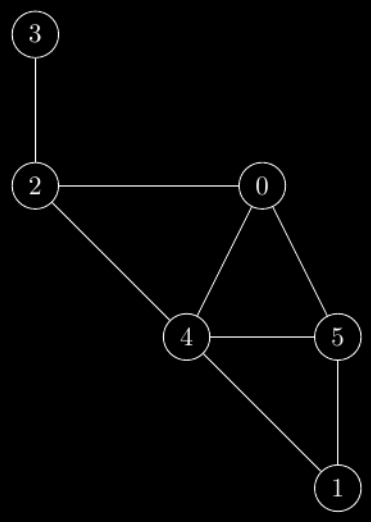
i) $T(n) = T(n-1) + c$, que es $O(n)$

ii) $T(n) = 4.T(n/2) + c$, que es $O(n^2)$

iii) $T(n) = 2.T(n-1) + c$, que es $O(2^n)$

iv) $T(n) = n.T(n-1) + c$, que es $O(n!)$

5 Grafos No Dirigidos 20%



a) (10%) ¿Cuál es un recorrido de *búsqueda primero en profundidad* del grafo anterior, si como nodo inicial se toma el nodo 1?

i) 1, 5, 0, 3, 2, 4

ii) 1, 4, 5, 0, 2, 3

iii) 1, 4, 0, 3, 5, 2

iv) 1, 5, 4, 0, 3, 2

b) (10%) ¿Cuál es un recorrido de *búsqueda primero en amplitud del grafo anterior*, si se toma como nodo inicial el nodo 1?

i) 1, 4, 5, 0, 2, 3

ii) 1, 5, 0, 2, 3, 4

iii) 1, 4, 2, 0, 3, 5

iv) 1, 3, 0, 4, 5, 2